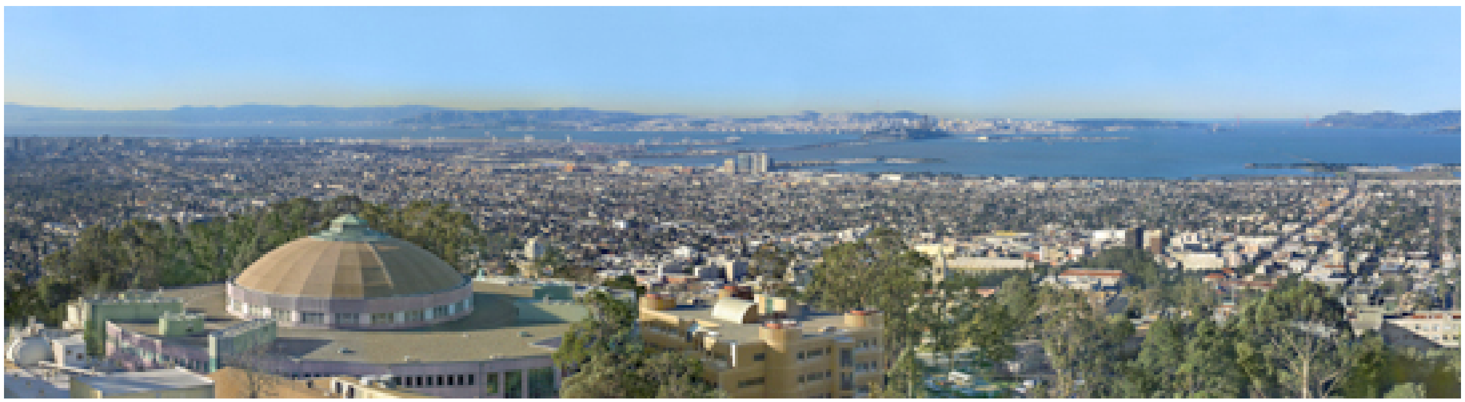
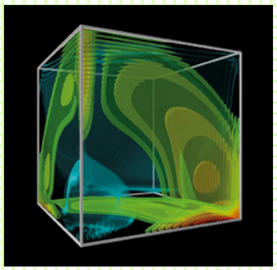
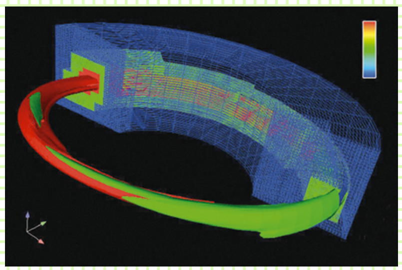
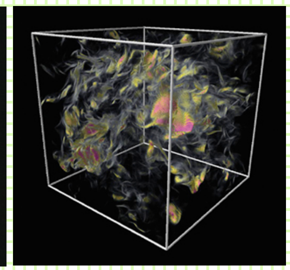
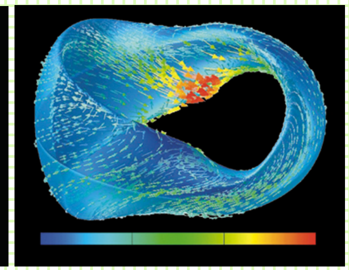
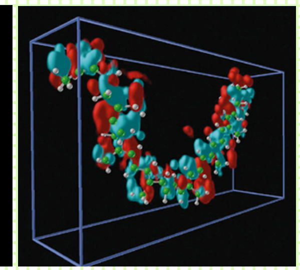
The DOE ACTS Collection

Fast and Robust Libraries for High Performance Computing



Tony Drummond
Lawrence Berkeley National Laboratory
LADrummond@lbl.gov

The Advanced CompuTational Software Collection Project

ACTS

Goal: The Advanced CompuTational Software Collection (ACTS) project makes reliable and efficient software tools more widely used, and more effective in solving the nation's engineering and scientific problems.

References:

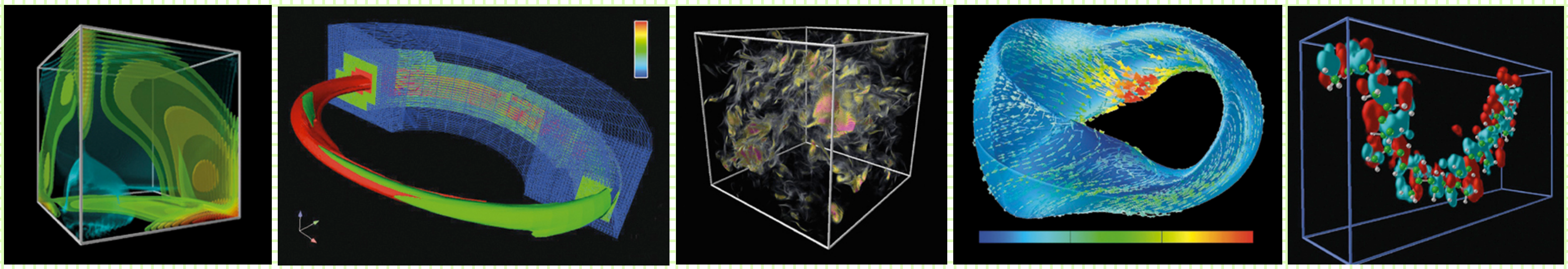
- L.A. Drummond, O. Marques: An Overview of the Advanced CompuTational Software (ACTS) Collection. ACM Transactions on Mathematical Software Vol. **31** pp. 282-301, 2005
- http://acts.nersc.gov

ACTS COLLECTION

BERKELEY LAB

Office of Science
U.S. DEPARTMENT OF ENERGY

The Advanced CompuTational Software Collection Project



Principal Investigators:

L. A. Drummond and O. A. Marques

Lawrence Berkeley National Laboratory

References:

- L.A. Drummond, O. Marques: An Overview of the Advanced CompuTational Software (ACTS) Collection. ACM Transactions on Mathematical Software Vol. **31** pp. 282-301, 2005
- http://acts.nersc.gov

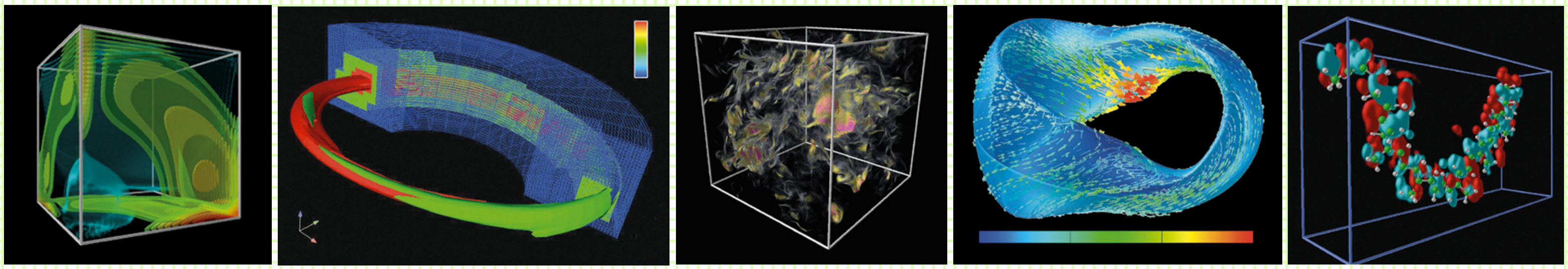
ACTS COLLECTION

Office of Science

U.S. DEPARTMENT OF ENERGY

BERKELEY LAB

The Advanced CompuTational Software Collection Project



acts-support@nersc.gov

References:

- L.A. Drummond, O. Marques: An Overview of the Advanced CompuTational Software (ACTS) Collection. ACM Transactions on Mathematical Software Vol. **31** pp. 282-301, 2005
- http://acts.nersc.gov

ACTS COLLECTION

BERKELEY LAB

Office of Science
U.S. DEPARTMENT OF ENERGY

The Core Activity Areas of the ACTS Collection Project

Long-Term Maintenance

Tools for Run-time Support:
- Scalable Debugging Tools
- Performance Monitoring Tools

Software Development Tools
Numerical Software
Library Optimization
Software Distribution
Software Dependency Graph:
- Platform, Basic and Interoperability

Independent Testing and Evaluation

Testing Platforms
Verification Engines
Computer Vendor Collaborations
Computational Sciences and Engineering Networking:
- Developers Exchange
- User Feedback
- Problem/Bug Tracking
- International Collaborations

DOE Leadership and Flagship Computing Facilities

ACTS TOOLS

Enabling Petascale

Outreach and Dissemination

ACTS Information Center:
- Guide to Available Services
- Technical Reports
- Newsgroups
- On-line Tutorials
- Uniform Tool Documentation
- Well Documented Examples for All Tools

Workshops
Short Courses and Coding Camps

High-Level User Support

High-Level User Interfaces

To Tool Users:
- Help with Tool Selection
- Help with Tool Utilization
- Help with Tool Installation
- Develop High-Level User Interfaces

To Tool Developers:
- Tool Long-term Maintenance Practices
- Tool Distribution Utilities, Licensing
- Tool Integration Mechanisms

$\cdots t_\infty$

Sustainable Software Support

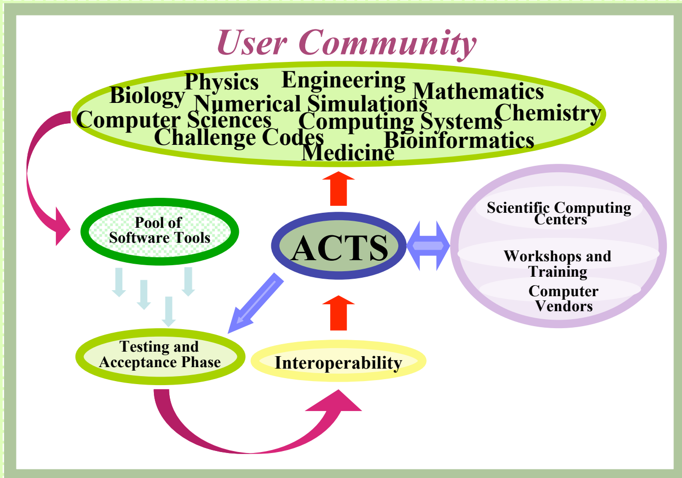
$\cdots t_\infty$

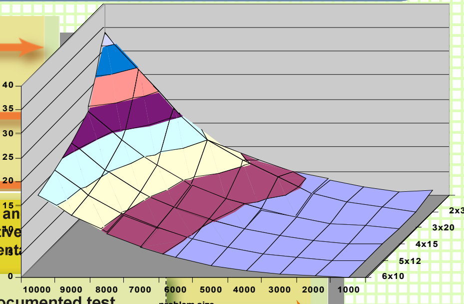
Berkeley Lab

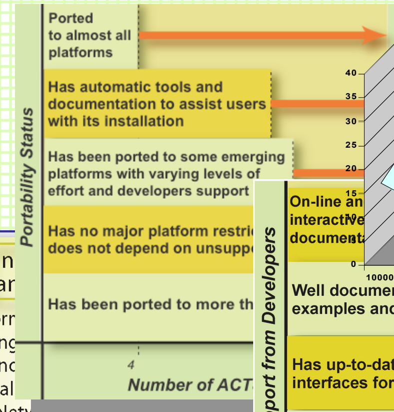
Office of Science
U.S. DEPARTMENT OF ENERGY

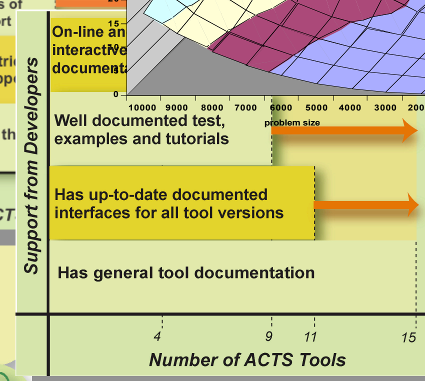
The U.S. DOE ACTS Collection Project

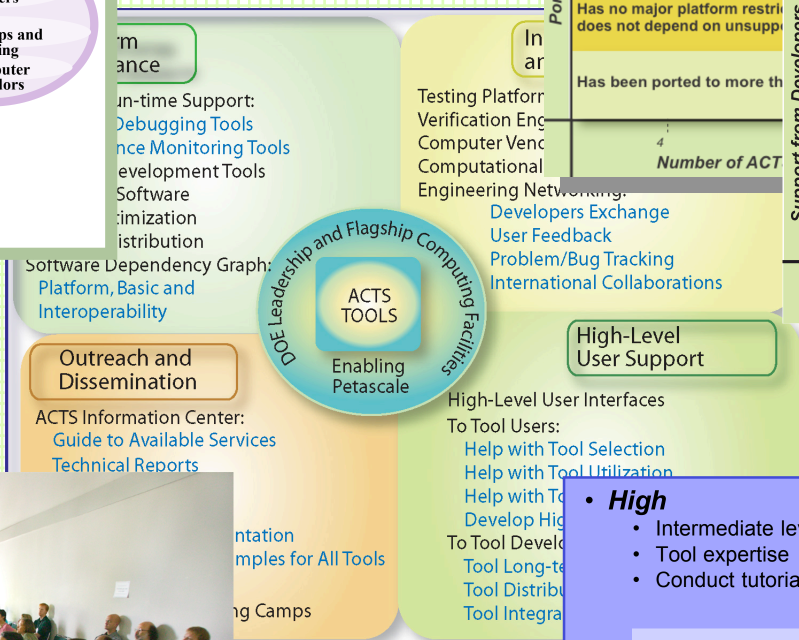
ACTS COLLECTION











- ***High***
  - Intermediate level
  - Tool expertise
  - Conduct tutorials
  - ***Intermediate***
    - Basic level
    - Higher level of support to users of the tool
    - ***Basic***
      - Help with installation
      - Basic knowledge of the tools
      - Compilation of user's reports

Office of Science
U.S. DEPARTMENT OF ENERGY

Tools In The ACTS Collection

Advanced CompuTational Software Collection (ACTS)

Funded by DOE/ASCR

LIBRARY DEVELOPMENT

NUMERICAL TOOLS

CODE DEVELOPMENT

RUN TIME SUPPORT

http://acts.nersc.gov

Category	Tool	Functionalities
Numerical	**Trilinos**	Algorithms for the iterative solution of large sparse linear systems.
	Hypre	Algorithms for the iterative solution of large sparse linear systems, intuitive grid-centric interfaces, and dynamic configuration of parameters.
	PETSc	Tools for the solution of PDEs that require solving large-scale, sparse linear and nonlinear systems of equations.
$Ax = b$ $Az = \lambda z$ $A = U\Sigma V^T$ PDEs ODEs M	**OPT++**	Object-oriented nonlinear optimization package.
	SUNDIALS	Solvers for the solution of systems of ordinary differential equations, nonlinear algebraic equations, and differential-algebraic equations.
	ScaLAPACK	Library of high performance dense linear algebra routines for distributed-memory message-passing.
	SLEPc	Software library for the solution of large sparse eigenproblems on parallel computers.
	SuperLU	General-purpose library for the direct solution of large, sparse, nonsymmetric systems of linear equations.
	TAO	Large-scale optimization software, including nonlinear least squares, unconstrained minimization, bound constrained optimization, and general nonlinear optimization.
Code Development	**Global Arrays**	Library for writing parallel programs that use large arrays distributed across processing nodes and that offers a shared-memory view of distributed arrays.
	Overture	Object-Oriented tools for solving computational fluid dynamics and combustion problems in complex geometries.
Run Time Support	**TAU**	Set of tools for analyzing the performance of C, C++, Fortran and Java programs.
Library Development	**ATLAS**	Tools for the automatic generation of optimized numerical software for modern computer architectures and compilers.

Functionality In The ACTS Collection

Computational Problem	Methodology	Algorithms	Library
Systems of Linear Equations	Direct Methods	LU Factorization	ScaLAPACK (dense) SuperLU (sparse)
		Cholesky Factorization	ScaLAPACK
		LDL^T (Tridiagonal matrices)	ScaLAPACK
		QR Factorization	ScaLAPACK
		QR with column pivoting	ScaLAPACK
		LQ factorization	ScaLAPACK

Functionality In The ACTS Collection

Computational Problem	Methodology	Algorithms	Library
Systems of Linear Equations (*cont..*)	Iterative Methods	Conjugate Gradient	AztecOO (Trilinos) PETSc
		GMRES	AztecOO PETSc Hypre
		CG Squared	AztecOO PETSc
		Bi-CG Stab	AztecOO PETSc
		Quasi-Minimal Residual (QMR)	AztecOO
		Transpose Free QMR	AztecOO PETSc

Functionality In The ACTS Collection

Computational Problem	Methodology	Algorithms	Library
Systems of Linear Equations *(cont..)*	Iterative Methods *(cont..)*	SYMMLQ	PETSc
		Precondition CG	AztecOO PETSc Hypre
		Richardson	PETSc
		Block Jacobi Preconditioner	AztecOO PETSc Hypre
		Point Jocobi Preconditioner	AztecOO
		Least Squares Polynomials	PETSc

Functionality In The ACTS Collection

Computational Problem	Methodology	Algorithms	Library
Systems of Linear Equations (*cont..*)	Iterative Methods (*cont..*)	SOR Preconditioning	PETSc
		Overlapping Additive Schwartz	PETSc
		Approximate Inverse	Hypre
		Sparse LU preconditioner	AztecOO PETSc Hypre
		Incomplete LU (ILU) preconditioner	AztecOO
		Least Squares Polynomials	PETSc
	MultiGrid (MG) Methods	MG Preconditioner	PETSc Hypre
		Algebraic MG	Hypre
		Semi-coarsening	Hypre

Functionality In The ACTS Collection

Computational Problem	Methodology	Algorithm	Library
Linear Least Squares Problems	Least Squares	$min_x \lVert b - Ax \rVert_2$	ScaLAPACK
	Minimum Norm Solution	$min_x \lVert x \rVert_2$	ScaLAPACK
	Minimum Norm Least Squares	$min_x \lVert b - Ax \rVert_2$ $min_x \lVert x \rVert_2$	ScaLAPACK
Standard Eigenvalue Problem	Symmetric Eigenvalue Problem	$Az = \lambda z$ For $A=A^H$ or $A=A^T$	ScaLAPACK (dense) SLEPc (sparse)
Singular Value Problem	Singular Value Decomposition	$A = U\Sigma V^T$ $A = U\Sigma V^H$	ScaLAPACK (dense) SLEPc (sparse)
Generalized Symmetric Definite Eigenproblem	Eigenproblem	$Az = \lambda Bz$ $ABz = \lambda z$ $BAz = \lambda z$	ScaLAPACK (dense) SLEPc (sparse)

Functionality In The ACTS Collection

Computational Problem	Methodology	Algorithm	Library
Non-Linear Equations	Newton Based	Line Search	PETSc
		Trust Regions	PETSc
		Pseudo-Transient Continuation	PETSc
		Matrix Free	PETSc

Functionality In The ACTS Collection

Computational Problem	Methodology	Algorithm	Library
Non-Linear Optimization	Newton Based	Newton	OPT++ TAO
		Finite-Difference Newton	OPT++ TAO
		Quasi-Newton	OPT++ TAO
		Non-linear Interior Point	OPT++ TAO
	CG	Standard Non-linear CG	OPT++ TAO
		Limited Memory BFGS	OPT++
		Gradient Projections	TAO
	Direct Search	No derivate information	OPT++

Functionality In The ACTS Collection

Computational Problem	Methodology	Algorithm	Library
Non-Linear Optimization (cont..)	Semismoothing	Feasible Semismooth	TAO
		Unfeasible semismooth	TAO
Ordinary Differential Equations	Integration	Adam-Moulton (Variable coefficient forms)	CVODE (SUNDIALS) CVODES
	Backward Differential Formula	Direct and Iterative Solvers	CVODE CVODES
Nonlinear Algebraic Equations	Inexact Newton	Line Search	KINSOL (SUNDIALS)
Differential Algebraic Equations	Backward Differential Formula	Direct and Iterative Solvers	IDA (SUNDIALS)

Functionality In The ACTS Collection

Computational Problem	Support	Techniques	Library
Writing Parallel Programs	Distributed Arrays	Shared-Memory	Global Arrays
		Distributed Memory	CUMULVS (viz) Globus (Grid)
		Grid Generation	OVERTURE
		Structured Meshes	CHOMBO (AMR) Hypre OVERTURE PETSc
		Semi-Structured Meshes	CHOMBO (AMR) Hypre OVERTURE

Functionality In The ACTS Collection

Computational Problem	Support	Technique	Library
Profiling	Algorithmic Performance	Automatic instrumentation	PETSc
		User Instrumentation	PETSc
	Execution Performance	Automatic Instrumentation	TAU
		User Instrumentation	TAU
Code Optimization	Library Installation	Linear Algebra Tuning	ATLAS
Interoperability	Code Generation	Language	BABEL
		Components	CCA

How Does One Use ACTS Tools?

```
 CALL BLACS_GET( -1, 0, ICTXT )
 CALL BLACS_GRIDINIT( ICTXT, 'Row-major', NPROW, NPCOL )
:
 CALL BLACS_GRIDINFO( ICTXT, NPROW, NPCOL, MYROW, MYCOL )
:
:
 CALL PDGESV( N, NRHS, A, IA, JA, DESCA, IPIV, B, IB, JB, DESCB,
 $                INFO )
```

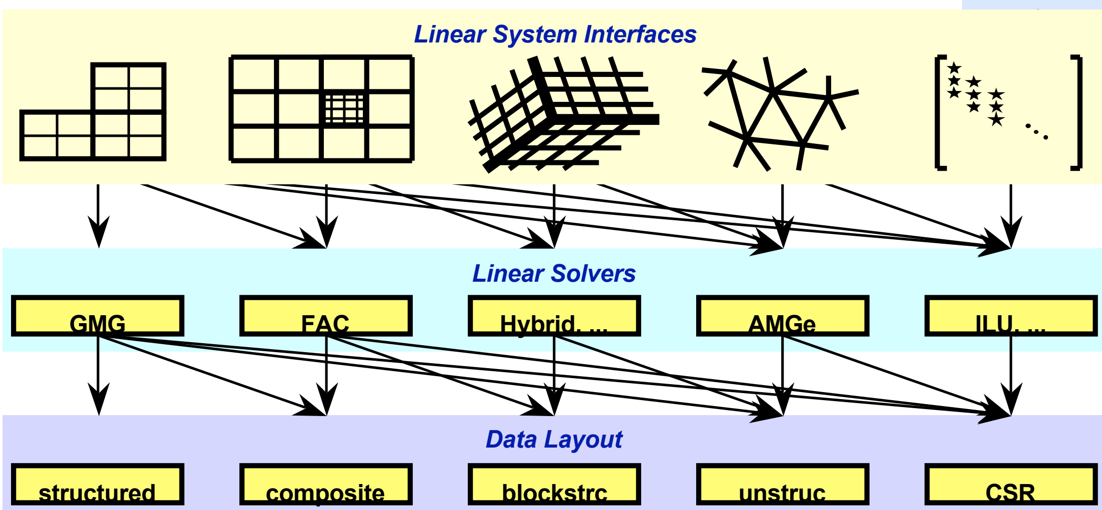
Language Calls

Command lines

- -ksp_type [cg,gmres,bcgs,tfqmr,…]
- -pc_type [lu,ilu,jacobi,sor,asm,…]

More advanced:

- -ksp_max_it <max_iters>
- -ksp_gmres_restart <restart>
- -pc_asm_overlap <overlap>
- -pc_asm_type <. . >

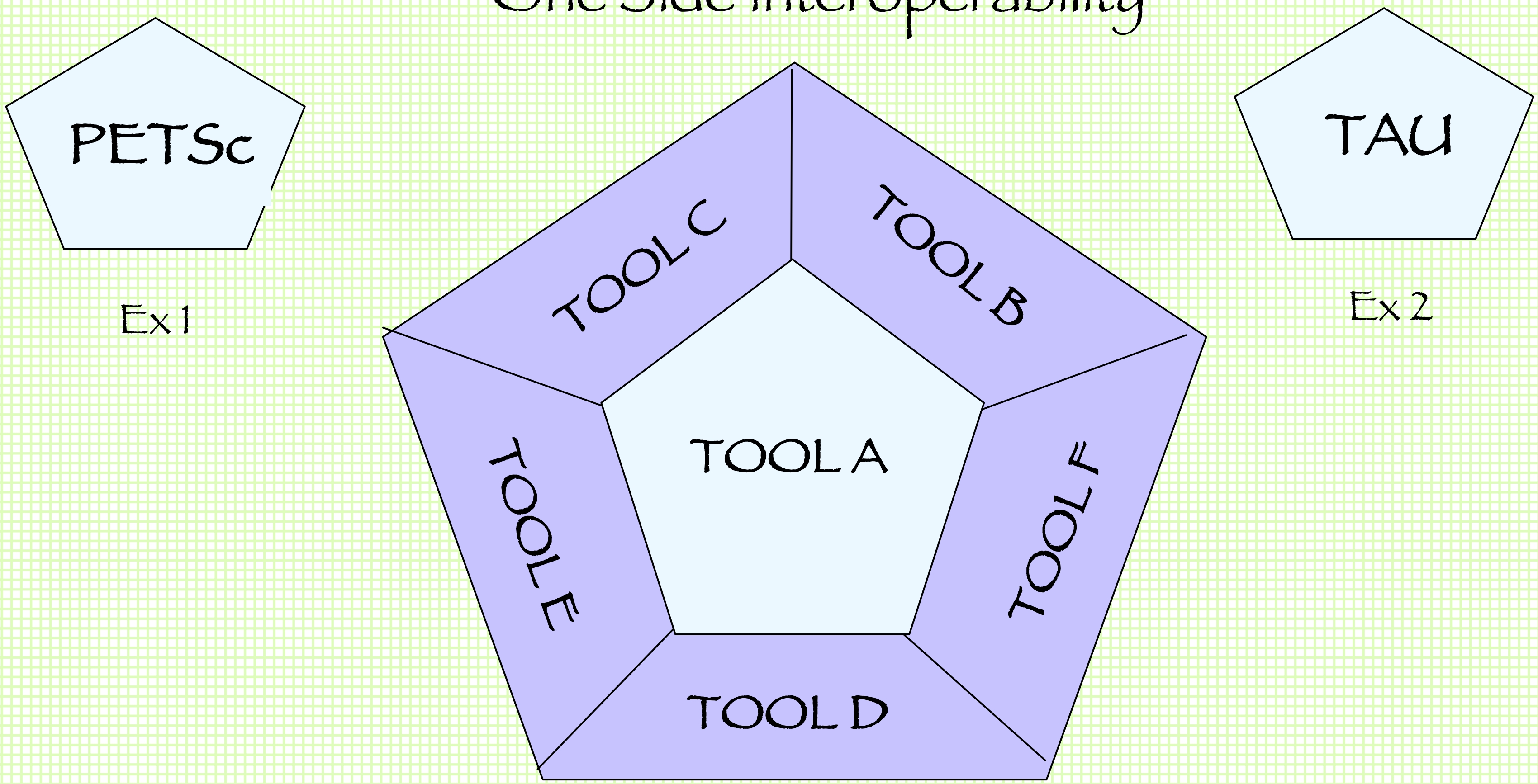


Problem Domain

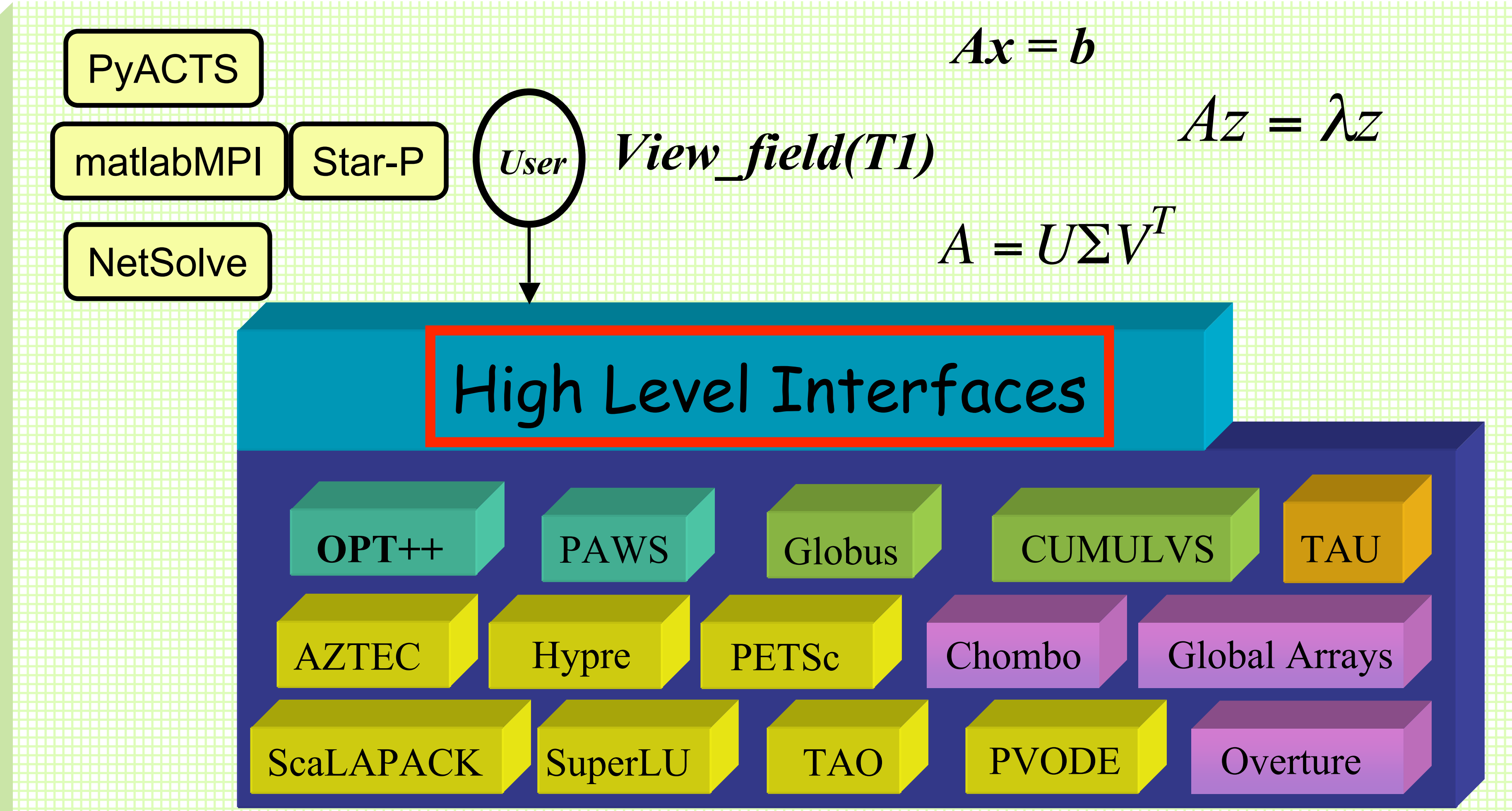
Office of Science
U.S. DEPARTMENT OF ENERGY
BERKELEY LAB

Tool to Tool Interoperability

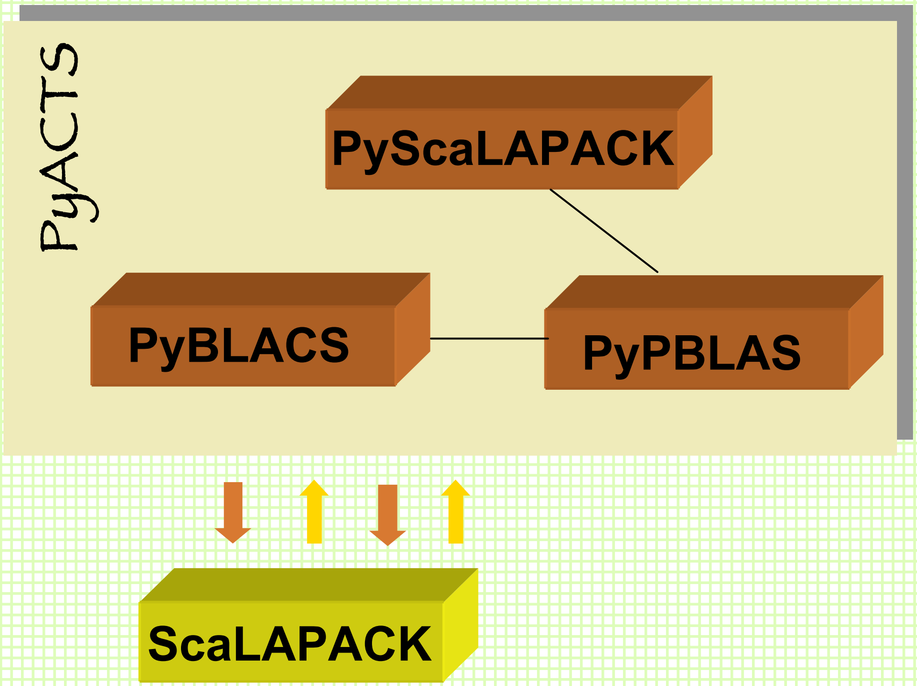
One Side Interoperability



High-level User Interfaces to the ACTS Collection



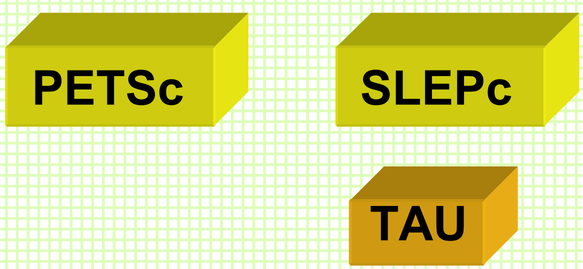
A Closer Look Into the ACTS Collection



$$\boldsymbol{Ax = b}$$

$$Az = \lambda z$$

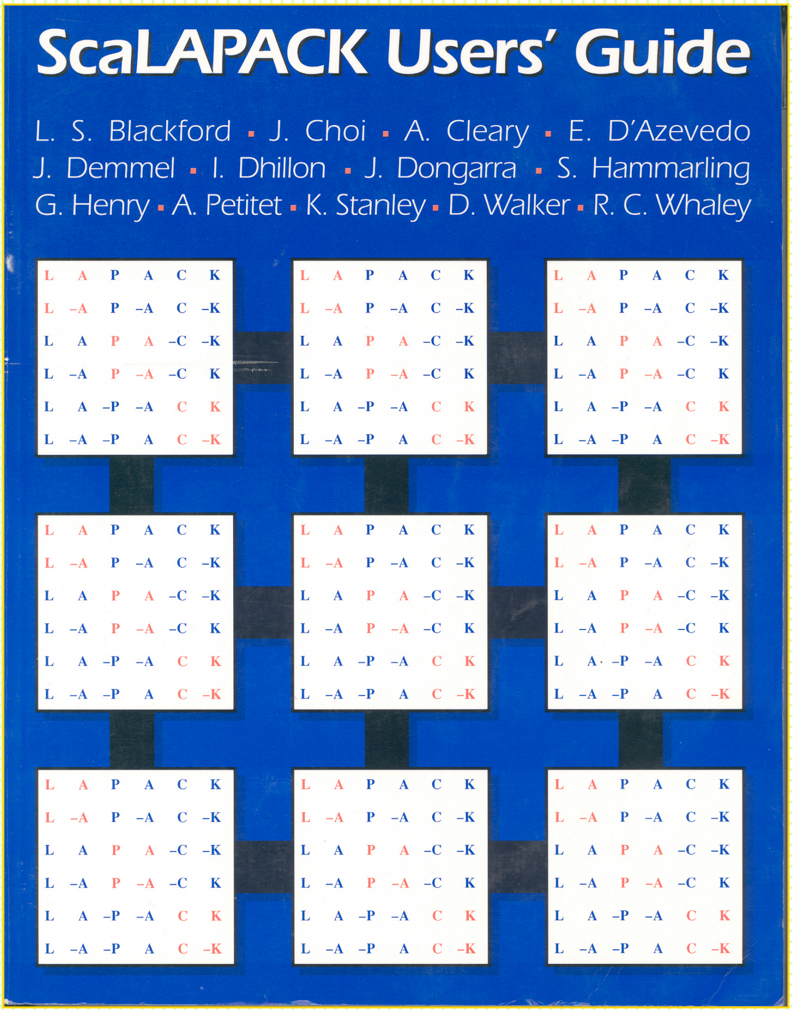
$$A = U\Sigma V^T$$



Dr. J. Roman

Dr. O. Marques

A Quick Introduction to ScaLAPACK



Team of Developers:

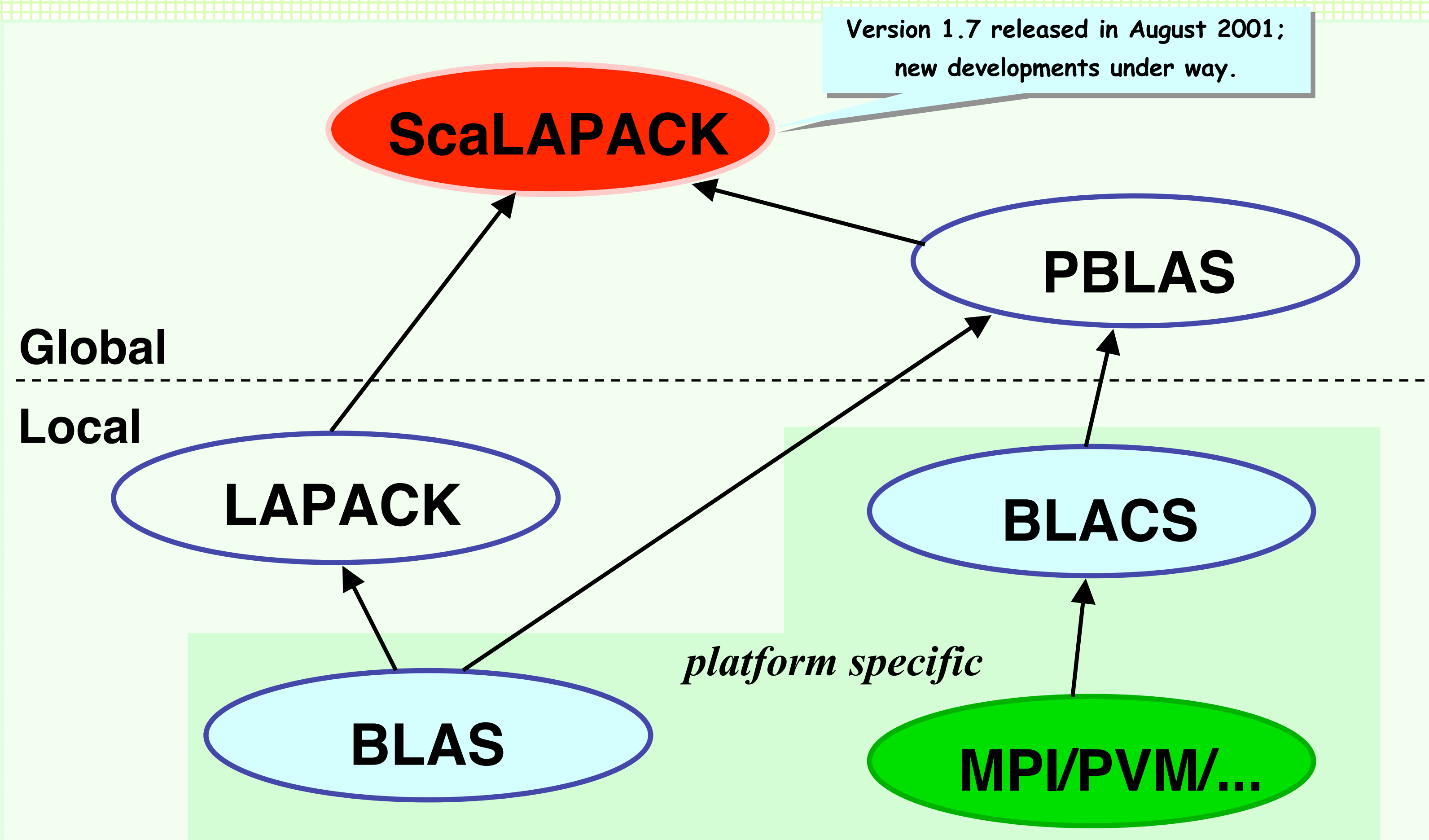
- Susan Blackford
- Jaeyoung Choi, Soongsil University
- Andy Cleary, LLNL
- Ed D'Azevedo, ORNL
- Jim Demmel, UCB
- Inderjit Dhillon, UT Austin
- Jack Dongarra, UTK
- Ray Fellers, LLNL
- Sven Hammarling, NAG
- Greg Henry, Intel
- Sherry Li, LBNL
- Osni Marques, LBNL
- Caroline Papadopoulos, UCSD
- Antoine Petitet, UTK
- Ken Stanley, UCB
- Francoise Tisseur, Manchester
- David Walker, Cardiff
- Clint Whaley, UTK
- Julien Langou, UTK

A Quick Introduction to ScaLAPACK

OUTLINE :

- ScaLAPACK: *software structure*
  - Basic Linear Algebra Subprograms (BLAS)
  - Linear Algebra PACKage (LAPACK)
  - Basic Linear Algebra Communication Subprograms (BLACS)
  - Parallel BLAS (PBLAS)
- ScaLAPACK: *details*
  - Data layout
  - Array descriptors
  - Error handling
  - Performance
- Examples

ScaLAPACK's Software Structure

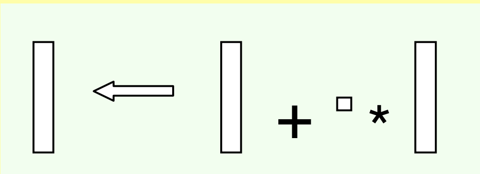


BLAS: Basic Linear Algebra Subroutines

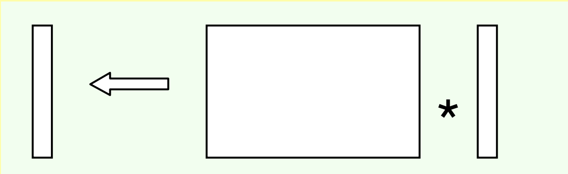
ACTS COLLECTION

BLAS LEVELS:

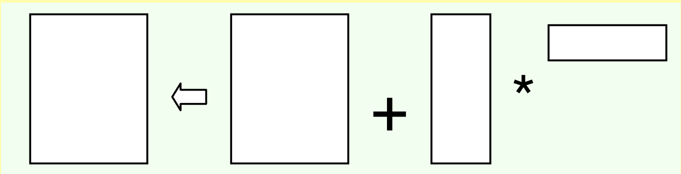
- Level 1 BLAS: vector-vector

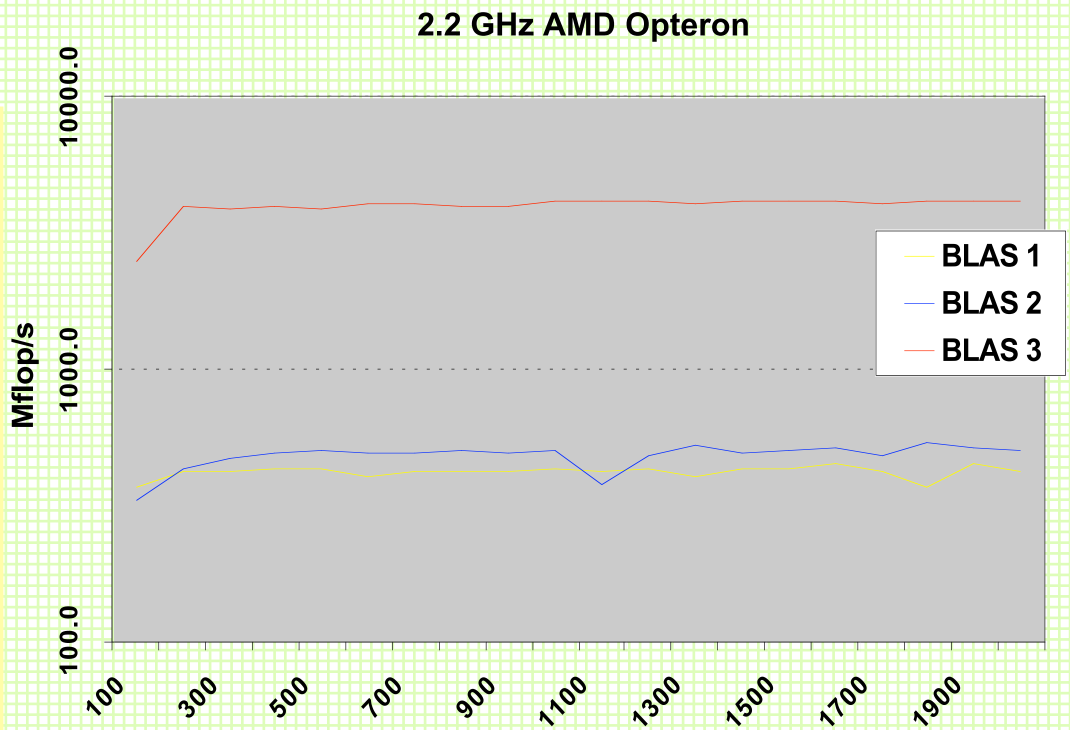


- Level 2 BLAS: matrix-vector



- Level 3 BLAS: matrix-matrix

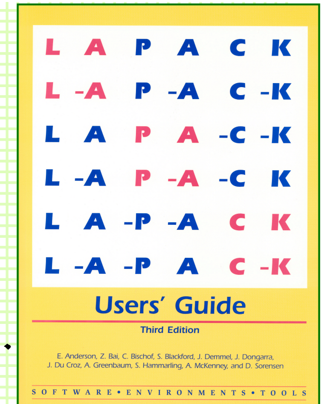




Design Considerations:

- Portability
- Performance: development of blocked algorithms is important for performance!

LAPACK: A Dense Linear Algebra Package



netlib.org

- Linear Algebra library written in Fortran 77 (Fortran 90)
- Combine algorithms from LINPACK and EISPACK into a single package.
- Efficient on a wide range of computers (RISC, Vector, SMPs).
- Built atop level 1, 2, and 3 BLAS Basic problems:
  - Linear systems: $Ax = b$
  - Least squares: $\min \|Ax - b\|_2$
  - Singular value decomposition: $A = U\Sigma V^T$
  - Eigenvalues and eigenvectors: $Az = \lambda z, \quad Az = \lambda Bz$
- LAPACK does not provide routines for structured problems or general sparse matrices (i.e. sparse storage formats such as compressed-row, -column, -diagonal, skyline ...).

BLACS:
Basic Linear Algebra Communication Subroutines

- Response to Message Passing based distributed communications
- Associate widely recognized mnemonic names with communication operations. This improves:
  - program readability
  - self-documenting quality of the code.
- Promote efficiency by identifying frequently occurring operations of linear algebra which can be optimized on various computers.

Basic Concepts of The BLACS Interface

- Promote efficiency by identifying common operations of linear algebra that can be optimized on various computers.
- Processes are embedded in a two-dimensional grid.

Example: a 3x4 grid

	0	1	2	3
0	0	1	2	3
1	4	5	6	7
2	8	9	10	11

- An operation which involves more than one sender and one receiver is called a *scoped operation.*

Scope	Meaning
Row	All processes in a process row participate.
Column	All processes in a process column participate.
All	All processes in the process grid participate.

BLACS Communication Routines

Send/Receive:

```
_xxSD2D(ICTXT,[UPLO,DIAG],M,N,A,LDA,RDEST,CDEST)
_xxRV2D(ICTXT,[UPLO,DIAG],M,N,A,LDA,RSRC,CSRC)
```

_ (Data type)	xx (Matrix type)
I: Integer, S: Real, D: Double Precision, C: Complex, Z: Double Complex.	GE: General rectangular matrix TR: Trapezoidal matrix

Broadcast:

```
_xxBS2D(ICTXT,SCOPE,TOP,[UPLO,DIAG],M,N,A,LDA)
_xxBR2D(ICTXT,SCOPE,TOP,[UPLO,DIAG],M,N,A,LDA,RSRC,CSRC)
```

SCOPE	TOP
'Row' 'Column' 'All'	' ' (default) 'Increasing Ring' '1-tree' ...

BLACS Context

- BLACS context ⇔ MPI communicator
- The BLACS context is the BLACS mechanism for partitioning communication space.
- A message in a context cannot be sent or received in another context.
- The context allows the user to
  - create arbitrary groups of processes
  - create multiple overlapping and/or disjoint grids
  - isolate each process grid so that grids do not interfere with each other

An Example Code Using BLACS

```
                  ⋮
*   Get system information
    CALL BLACS_PINFO( IAM, NPROCS )
    ⋮
*   Get default system context
    CALL BLACS_GET( 0, 0, ICTXT )
    ⋮
*   Define 1 x (NPROCS/2+1) process grid
    NPROW = 1
    NPCOL = NPROCS / 2 + 1
    CALL BLACS_GRIDINIT( ICTXT, 'Row', NPROW, NPCOL )
    CALL BLACS_GRIDINFO( ICTXT, NPROW, NPCOL, MYROW, MYCOL )
*   If I'm not in the grid, go to end of program
    IF( MYROW.NE.-1 ) THEN
      IF( MYROW.EQ.0 .AND. MYCOL.EQ.0 ) THEN
        CALL DGESD2D( ICTXT, 5, 1, X, 5, 1, 0 )
      ELSE IF( MYROW.EQ.1 .AND. MYCOL.EQ.0 ) THEN
        CALL DGERV2D( ICTXT, 5, 1, Y, 5, 0, 0 )
      END IF
       ⋮
      CALL BLACS_GRIDEXIT( ICTXT )
    END IF
    ⋮
    CALL BLACS_EXIT( 0 )
    END
```

(out) uniquely identifies each process
(out) number of processes available

(in) integer handle indicating the context
(in) use (default) system context
(out) BLACS context

(output) process row and column coordinate

send X to process (1,0)

receive X from process (0,0)

leave context

exit from the BLACS

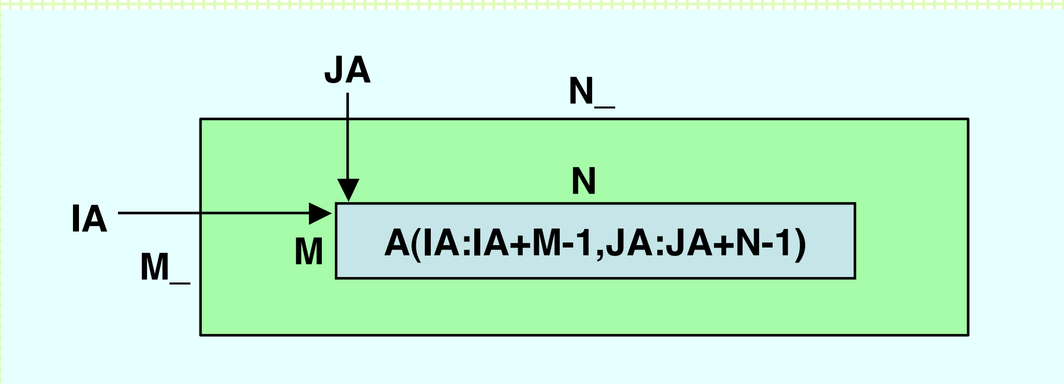
PBLAS: Parallel BLAS

- Similar to the BLAS in portability, functionality and naming.
- Built atop the BLAS and BLACS
- Provide global view of matrix

```
CALL DGEXXX( M, N, A( IA, JA ), LDA, ... )
```
BLAS

```
CALL PDGEXXX( M, N, A, IA, JA, DESCA, ... )
```
PBLAS

Array descriptor (to be reviewed later)

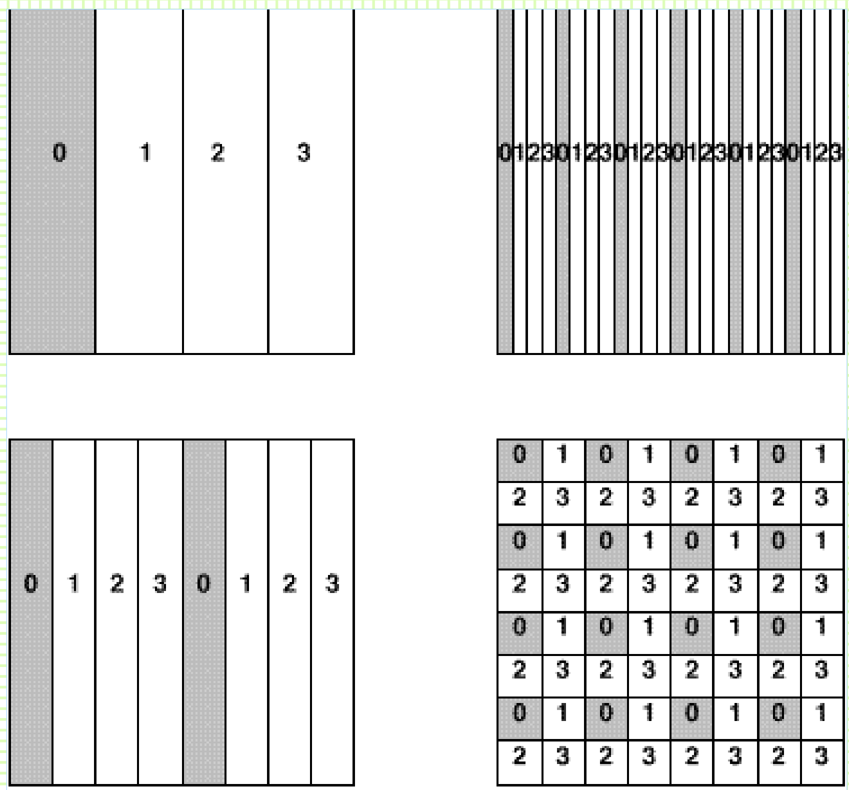


ScaLAPACK Design Goals

- Efficiency
  - Optimized computation and communication engines
  - Block-partitioned algorithms (Level 3 BLAS) for good node performance
- Reliability
  - Whenever possible, use LAPACK algorithms and error bounds.
- Scalability
  - As the problem size and number of processors grow
  - Replace LAPACK algorithm that did not scale (new ones into LAPACK)
- Portability
  - Isolate machine dependencies to BLAS and the BLACS
- Flexibility
  - Modularity: build rich set of linear algebra tools (BLAS, BLACS, PBLAS)
- Ease-of-Use
  - Calling interface similar to LAPACK

ScaLAPACK: Data Layouts

- 1D block and column distributions
- 1D block-cycle column and 2D block-cyclic distribution
- 2D block-cyclic distribution used in ScaLAPACK for dense matrices



How does 2D Block Cyclic Distribution Work

5x5 matrix partitioned in 2x2 blocks

$$
\left(\begin{array}{cc:cc:c}
a_{11} & a_{12} & a_{13} & a_{14} & a_{15} \\
a_{21} & a_{22} & a_{23} & a_{24} & a_{25} \\
\hdashline
a_{31} & a_{32} & a_{33} & a_{34} & a_{35} \\
a_{41} & a_{42} & a_{43} & a_{44} & a_{45} \\
\hdashline
a_{51} & a_{52} & a_{53} & a_{54} & a_{55}
\end{array}\right)
$$

2x2 process grid point of view

Process 0

a_{11}	a_{12}	a_{15}
a_{21}	a_{22}	a_{25}
a_{51}	a_{52}	a_{55}

Process 1

a_{13}	a_{14}
a_{23}	a_{24}
a_{53}	a_{54}

Process 2

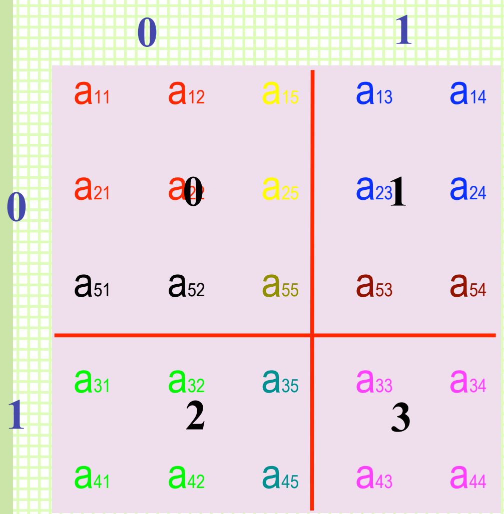
a_{31}	a_{32}	a_{35}
a_{41}	a_{42}	a_{45}

Process 3

a_{33}	a_{34}
a_{43}	a_{44}

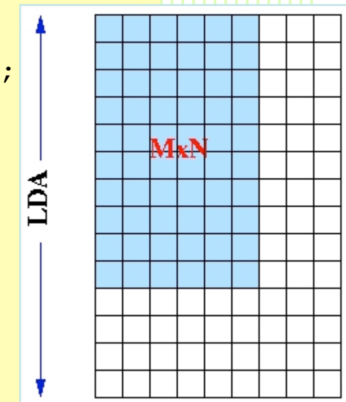
An Example of 2D Block Cyclic Distribution

$$
\begin{bmatrix}
1.1 & 1.2 & 1.3 & 1.4 & 1.5 \\
-2.1 & 2.2 & 2.3 & 2.4 & 2.5 \\
-3.1 & -3.2 & 3.3 & 3.4 & 3.5 \\
-4.1 & -4.2 & -4.3 & 4.4 & 4.5 \\
-5.1 & -5.2 & -5.3 & -5.4 & 5.5
\end{bmatrix}
$$




```
:

CALL BLACS_GRIDINFO( ICTXT, NPROW, NPCOL, MYROW, MYCOL )

IF      ( MYROW.EQ.0 .AND. MYCOL.EQ.0 ) THEN
          A(1) = 1.1; A(2) = -2.1; A(3) = -5.1;
          A(1+LDA) = 1.2; A(2+LDA) = 2.2; A(3+LDA) = -5.2;
          A(1+2*LDA) = 1.5; A(2+3*LDA) = 2.5; A(3+4*LDA) = -5.5;
ELSE IF ( MYROW.EQ.0 .AND. MYCOL.EQ.1 ) THEN
          A(1) = 1.3; A(2) = 2.3; A(3) = -5.3;
          A(1+LDA) = 1.4; A(2+LDA) = 2.4; A(3+LDA) = -5.4;
ELSE IF ( MYROW.EQ.1 .AND. MYCOL.EQ.0 ) THEN
          A(1) = -3.1; A(2) = -4.1;
          A(1+LDA) = -3.2; A(2+LDA) = -4.2;
          A(1+2*LDA) = 3.5; A(2+3*LDA) = 4.5;
ELSE IF ( MYROW.EQ.1 .AND. MYCOL.EQ.1 ) THEN
          A(1) = 3.3; A(2) = -4.3;
          A(1+LDA) = 3.4; A(2+LDA) = 4.4;
END IF

  :

CALL PDGESVD( JOBU, JOBVT, M, N, A, IA, JA, DESCA, S, U, IU,
              JU, DESCU, VT, IVT, JVT, DESCVT, WORK, LWORK,
              INFO )
  :
```

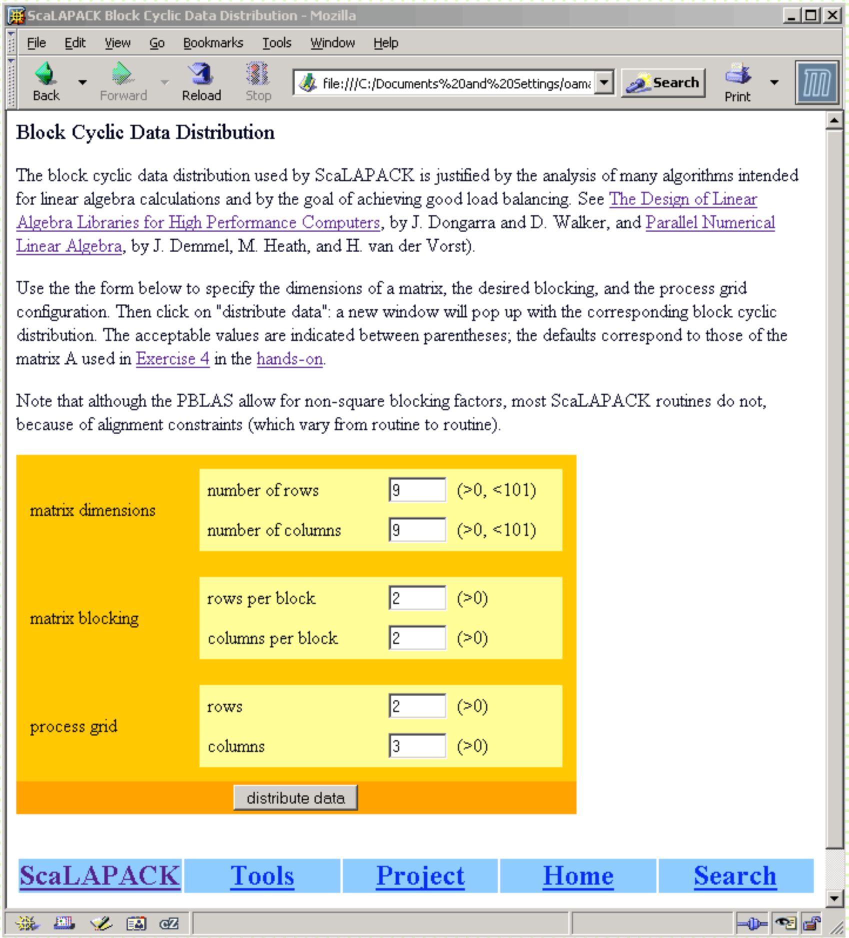
LDA is the leading dimension of the local array



Why the headache of 2D block Cyclic Distribution?

- Ensures good load balance → performance and scalability (analysis of many algorithms to justify this layout).
- Encompasses a large number of data distribution schemes (but not all).
- Needs redistribution routines to go from one distribution to the other.
- See http://acts.nersc.gov/scalapack/hands-on/datadist.html

AID: http://acts.nersc.gov/scalapack/hands-on/datadist.html



ScaLAPACK Block Cyclic Data Distribution - Mozilla

File Edit View Go Bookmarks Tools Window Help

Back Forward Reload Stop file:///C:/Documents%20and%20Settings/oam: Search Print

Block Cyclic Data Distribution

The block cyclic data distribution used by ScaLAPACK is justified by the analysis of many algorithms intended for linear algebra calculations and by the goal of achieving good load balancing. See The Design of Linear Algebra Libraries for High Performance Computers, by J. Dongarra and D. Walker, and Parallel Numerical Linear Algebra, by J. Demmel, M. Heath, and H. van der Vorst).

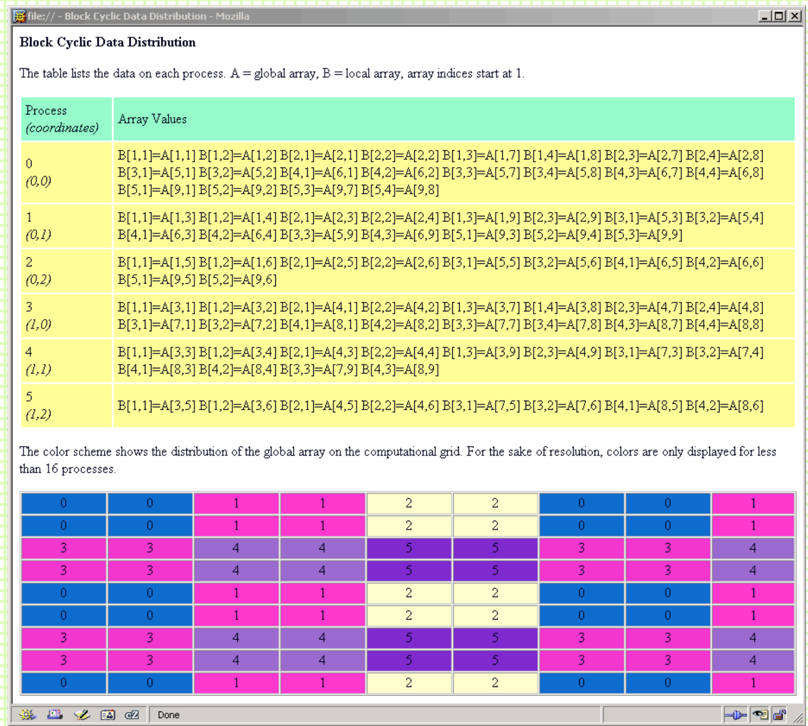
Use the the form below to specify the dimensions of a matrix, the desired blocking, and the process grid configuration. Then click on "distribute data": a new window will pop up with the corresponding block cyclic distribution. The acceptable values are indicated between parentheses; the defaults correspond to those of the matrix A used in Exercise 4 in the hands-on.

Note that although the PBLAS allow for non-square blocking factors, most ScaLAPACK routines do not, because of alignment constraints (which vary from routine to routine).

matrix dimensions	number of rows	9	(>0, <101)
	number of columns	9	(>0, <101)
matrix blocking	rows per block	2	(>0)
	columns per block	2	(>0)
process grid	rows	2	(>0)
	columns	3	(>0)

distribute data

ScaLAPACK | Tools | Project | Home | Search



file:// - Block Cyclic Data Distribution - Mozilla

Block Cyclic Data Distribution

The table lists the data on each process. A = global array, B = local array, array indices start at 1.

Process *(coordinates)*	Array Values
0 *(0,0)*	B[1,1]=A[1,1] B[1,2]=A[1,2] B[2,1]=A[2,1] B[2,2]=A[2,2] B[1,3]=A[1,7] B[1,4]=A[1,8] B[2,3]=A[2,7] B[2,4]=A[2,8] B[3,1]=A[5,1] B[3,2]=A[5,2] B[4,1]=A[6,1] B[4,2]=A[6,2] B[3,3]=A[5,7] B[3,4]=A[5,8] B[4,3]=A[6,7] B[4,4]=A[6,8] B[5,1]=A[9,1] B[5,2]=A[9,2] B[5,3]=A[9,7] B[5,4]=A[9,8]
1 *(0,1)*	B[1,1]=A[1,3] B[1,2]=A[1,4] B[2,1]=A[2,3] B[2,2]=A[2,4] B[1,3]=A[1,9] B[2,3]=A[2,9] B[3,1]=A[5,3] B[3,2]=A[5,4] B[4,1]=A[6,3] B[4,2]=A[6,4] B[3,3]=A[5,9] B[4,3]=A[6,9] B[5,1]=A[9,3] B[5,2]=A[9,4] B[5,3]=A[9,9]
2 *(0,2)*	B[1,1]=A[1,5] B[1,2]=A[1,6] B[2,1]=A[2,5] B[2,2]=A[2,6] B[3,1]=A[5,5] B[3,2]=A[5,6] B[4,1]=A[6,5] B[4,2]=A[6,6] B[5,1]=A[9,5] B[5,2]=A[9,6]
3 *(1,0)*	B[1,1]=A[3,1] B[1,2]=A[3,2] B[2,1]=A[4,1] B[2,2]=A[4,2] B[1,3]=A[3,7] B[1,4]=A[3,8] B[2,3]=A[4,7] B[2,4]=A[4,8] B[3,1]=A[7,1] B[3,2]=A[7,2] B[4,1]=A[8,1] B[4,2]=A[8,2] B[3,3]=A[7,7] B[3,4]=A[7,8] B[4,3]=A[8,7] B[4,4]=A[8,8]
4 *(1,1)*	B[1,1]=A[3,3] B[1,2]=A[3,4] B[2,1]=A[4,3] B[2,2]=A[4,4] B[1,3]=A[3,9] B[2,3]=A[4,9] B[3,1]=A[7,3] B[3,2]=A[7,4] B[4,1]=A[8,3] B[4,2]=A[8,4] B[3,3]=A[7,9] B[4,3]=A[8,9]
5 *(1,2)*	B[1,1]=A[3,5] B[1,2]=A[3,6] B[2,1]=A[4,5] B[2,2]=A[4,6] B[3,1]=A[7,5] B[3,2]=A[7,6] B[4,1]=A[8,5] B[4,2]=A[8,6]

The color scheme shows the distribution of the global array on the computational grid. For the sake of resolution, colors are only displayed for less than 16 processes.

0	0	1	1	2	2	0	0	1
0	0	1	1	2	2	0	0	1
3	3	4	4	5	5	3	3	4
3	3	4	4	5	5	3	3	4
0	0	1	1	2	2	0	0	1
0	0	1	1	2	2	0	0	1
3	3	4	4	5	5	3	3	4
3	3	4	4	5	5	3	3	4
0	0	1	1	2	2	0	0	1

Done

ScaLAPACK: Array Descriptors

- Each global data object is assigned an *array descriptor*.
- The *array descriptor*:
  - Contains information required to establish mapping between a global array entry and its corresponding process and memory location (uses concept of BLACS context).
  - Is differentiated by the DTYPE_ (first entry) in the descriptor.
  - Provides a flexible framework to easily specify additional data distributions or matrix types.
- User must distribute all global arrays prior to the invocation of a ScaLAPACK routine, for example:
  - Each process generates its own submatrix.
  - One processor reads the matrix from a file and send pieces to other processors (may require message-passing for this).

Array Descriptor for Dense Matrices

DESC_()	Symbolic Name	Scope	Definition
1	DTYPE_A	(global)	Descriptor type DTYPE_A=1 for dense matrices.
2	CTXT_A	(global)	BLACS context handle.
3	M_A	(global)	Number of rows in global array A.
4	N_A	(global)	Number of columns in global array A.
5	MB_A	(global)	Blocking factor used to distribute the rows of array A.
6	NB_A	(global)	Blocking factor used to distribute the columns of array A.
7	RSRC_A	(global)	Process row over which the first row of the array A is distributed.
8	CSRC_A	(global)	Process column over which the first column of the array A is distributed.
9	LLD_A	(local)	Leading dimension of the local array.

Array Descriptor for Narrow Band Matrices

DESC_( )	Symbolic Name	Scope	Definition
1	DTYPE_A	(global)	Descriptor type DTYPE_A=501 for 1 x Pc process grid for band and tridiagonal matrices block-column distributed.
2	CTXT_A	(global)	BLACS context handle.
3	N_A	(global)	Number of columns in global array A.
4	NB_A	(global)	Blocking factor used to distribute the columns of array A.
5	CSRC_A	(global)	Process column over which the first column of the array A is distributed.
6	LLD_A	(local)	Leading dimension of the local array. For the tridiagonal subroutines, this entry is ignored.
7	–	–	Unused, reserved.

Array Descriptor for Right Hand Sides for Narrow Band Linear Solvers

ACTS COLLECTION

DESC_( )	Symbolic Name	Scope	Definition
1	DTYPE_B	(global)	Descriptor type DTYPE_B=502 for Pr x 1 process grid for block-row distributed matrices
2	CTXT_B	(global)	BLACS context handle
3	M_B	(global)	Number of rows in global array B
4	MB_B	(global)	Blocking factor used to distribute the rows of array B
5	RSRC_B	(global)	Process row over which the first row of the array B is distributed
6	LLD_B	(local)	Leading dimension of the local array. For the tridiagonal subroutines, this entry is ignored
7	–	–	Unused, reserved

ScaLAPACK Functionality

$Ax = b$	Driver type: Simple	Driver type: Expert	Factor	Solve	Inversion	Conditioning estimator	Iterative Refinement
Triangular	×			×	×	×	×
SPD	×	×	×	×	×	×	×
SPD Banded	×		×	×			
SPD Tridiagonal	×		×	×			
General	×	×	×	×	×	×	×
General Banded	×		×	×			
General Tridiagonal	×		×	×			
Least Squares	×		×	×			
GQR			×				
GRQ			×				
$Ax = \lambda x$ or $Ax = \lambda Bx$	**Simple**	**Expert**	**Reduce**	**Solve**			
Symmetric	×	×	×	×			
General	×	×	×	×			
Generalized BSPD	×		×	×			
SVD			×	×			

ScaLAPACK: Error Handling

- Driver and computational routines perform *global* and *local* input error-checking.
  - Global checking → synchronization
  - Local checking → validity
- No input error-checking is performed on the auxiliary routines.
- If an error is detected in a PBLAS or BLACS routine program execution stops.

ScaLAPACK: Debugging Hints

- Look at ScaLAPACK example programs.
- Always check the value of INFO on exit from a ScaLAPACK routine.
- Query for size of workspace, LWORK = –1.
- Link to the Debug Level 1 BLACS (specified by BLACSDBGLVL=1 in Bmake.inc).
- Consult errata files on *netlib*:

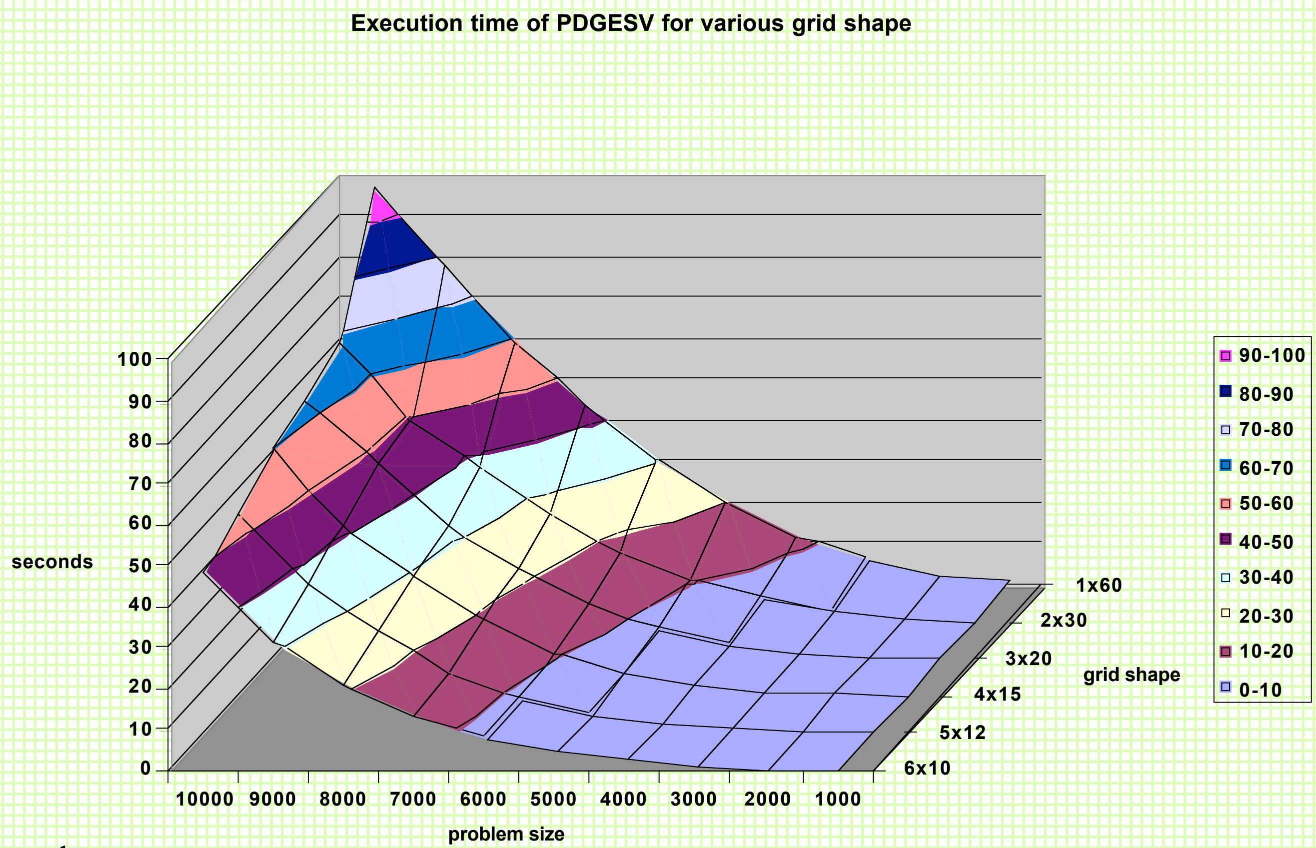
  http://www.netlib.org/scalapack/errata.scalapack

  http://www.netlib.org/blacs/errata.blacs

ScaLAPACK Performance

- The algorithms implemented in ScaLAPACK are scalable in the sense that the parallel efficiency is an increasing function of N^2/P (problem size per node).
- Maintaining memory use per node constant allows efficiency to be maintained (in practice, a slight degradation is acceptable).
- Use efficient machine-specific BLAS (not the Fortran 77 source code available in *http://www.netlib.gov*) and BLACS (nondebug installation).
- On a distributed-memory computer:
  - Use the right number of processors
    - Rule of thumb: $P=MxN/10^6$ for an MxN matrix, which provides a local matrix of size approximately 1000-by-1000.
    - Do not try to solve a small problem on too many processors.
    - Do not exceed the physical memory.
  - Use an efficient data distribution.
    - Block size (i.e., MB,NB) = 64.
    - Square processor grid: Prow = Pcolumn.

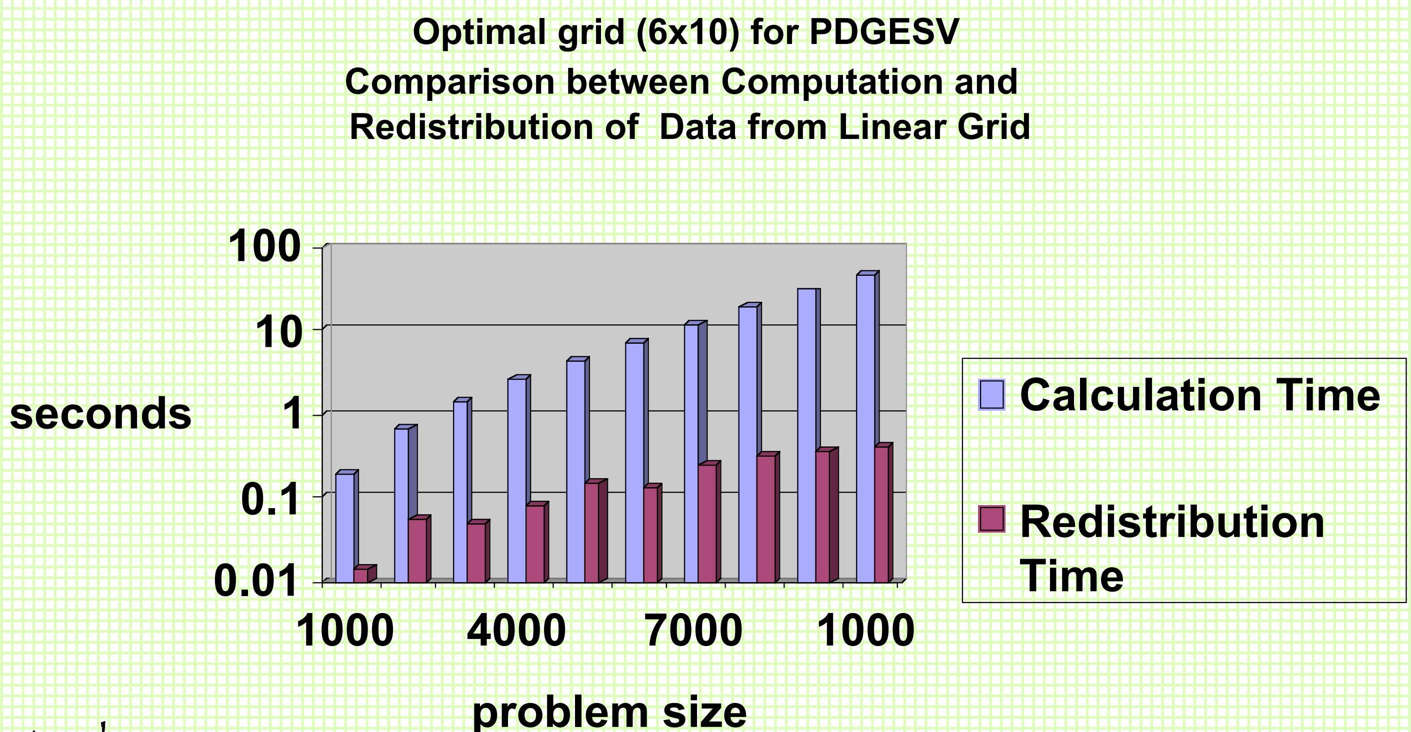
ScaLAPACK Performance: Varying Proc Grid Size



Times obtained on:

60 processors, Dual AMD Opteron 1.4GHz Cluster with Myrinet Interconnect, 2GB Memory

ScaLAPACK Performance: Computation vs. Communication



Times obtained on:

60 processors, Dual AMD Opteron 1.4GHz Cluster w/Myrinet Interconnect 2GB Memory

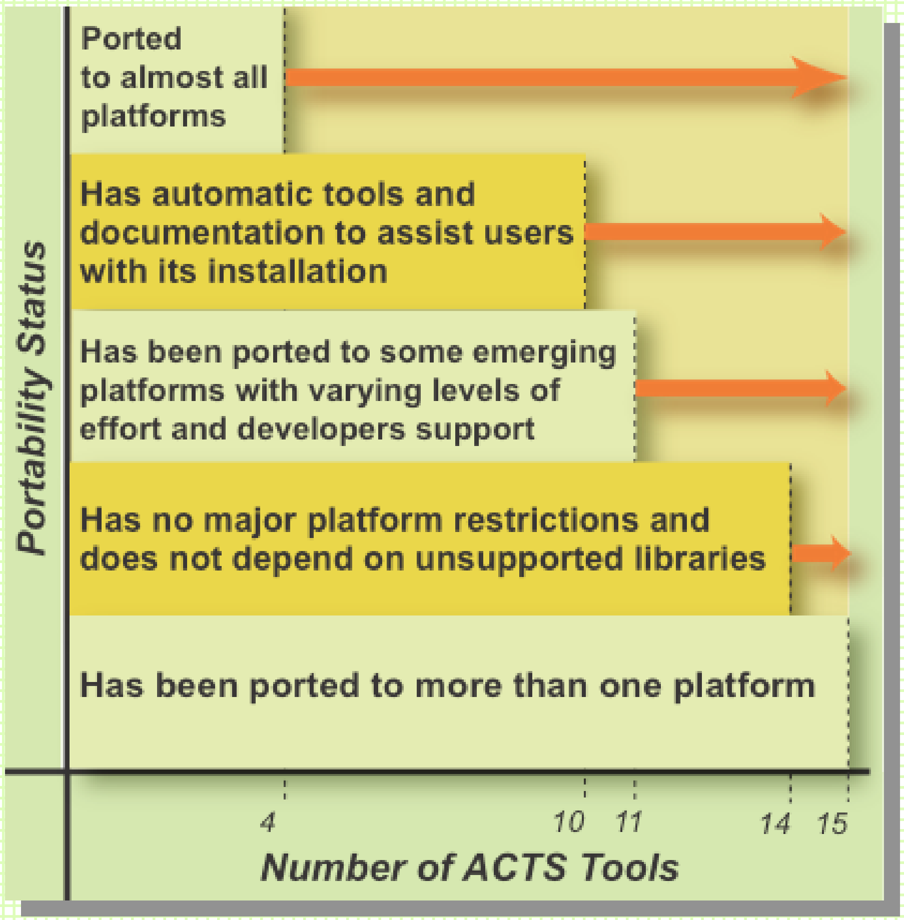
Commercial use of ScaLAPACK

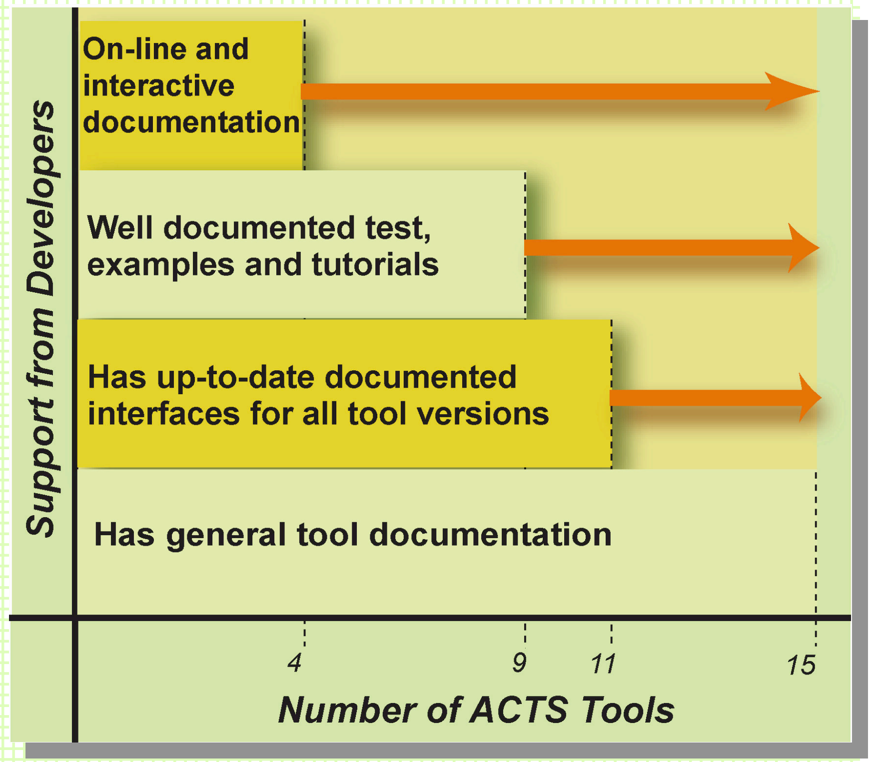
ScaLAPACK has been incorporated in the following commercial packages:

- Fujitsu
- Hewlett-Packard
- Hitachi
- IBM Parallel ESSL
- NAG Numerical Library
- Cray LIBSCI
- NEC Scientific Software Library
- Sun Scientific Software Library
- Visual Numerics (IMSL)

SUMMARY

Software Sustainability Requirement





SUMMARY

min[*time_to_first_solution*] (prototype) PyACTS

min[*time_to_solution*] (production)

- Outlive Complexity
  - Increasingly sophisticated models
  - Model coupling
  - Interdisciplinary

(Software Evolution)

- Sustained Performance
  - Increasingly complex algorithms
  - Increasingly diverse architectures

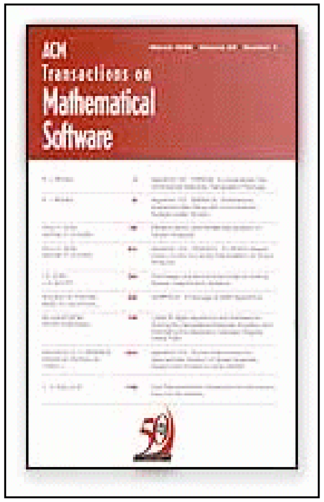
(Long-term deliverables)

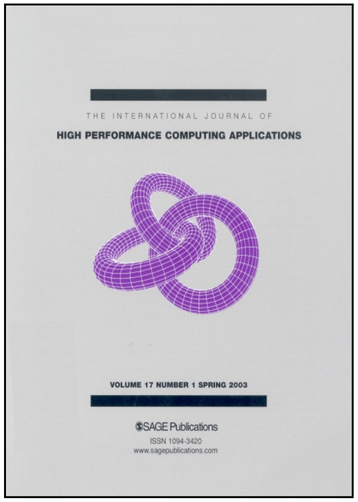
min[*software-development-cost*]

max[*software_life*] and max[*resource_utilization*]

References

- ScaLAPACK and PyACTS hands-on this week
- PETSc and SLEPc tutorials this week
- ACTS Information Center: http://acts.nersc.gov
- Two Journal Issues dedicated to ACTS





- Eighth ACTS Collection Workshop, August 21-24, 2007